Traces of Term-Automatic Graphs

Antoine Meyer

Traces of term-automatic graphs

Antoine Meyer

LIAFA – Université Paris Diderot – Paris 7
Case 7014, 2 place Jussieu, 75251 Paris Cedex 05, France
ameyer@liafa.jussieu.fr

Abstract. In formal language theory, many families of languages are defined using grammars or finite acceptors like pushdown automata and Turing machines. For instance, context-sensitive languages are the languages generated by growing grammars, or equivalently those accepted by Turing machines whose work tape's size is proportional to that of their input. A few years ago, a new characterisation of context-sensitive languages as the sets of traces, or path labels, of rational graphs (infinite graphs defined by sets of finite-state transducers) was established.
We investigate a similar characterisation in the more general framework of graphs defined by term transducers. In particular, we show that the languages of term-automatic graphs between regular sets of vertices coincide with the languages accepted by alternating linearly bounded Turing machines. As a technical tool, we also introduce an arborescent variant of tiling systems, which provides yet another characterisation of these languages.

Introduction

In classical language theory, context-sensitive languages, one of the families of the Chomsky hierarchy [Cho59], are defined as the languages generated by growing grammars. They were later characterised as the languages accepted by linearly space-bounded Turing machines [Kur64], i.e. Turing machines whose runs on any input word of length n use at most $k \cdot n$ work tape cells, for some constant k. In [LS97], it was shown that context-sensitive languages also coincide with the languages accepted by bounded tiling systems.

In 2001, [MS01] provided yet another characterisation of this family as the set of path languages of rational graphs [Mor00], i.e. infinite graphs whose vertices are words and whose sets of edges are defined by finite transducers. This result was later extended in [Ris02] to the more restricted family of automatic graphs (Cf. [KN95]), and even to synchronous rational graphs when an infinite number of initial and final vertices are considered (see also [MR05]). In a way, this provides a "forward", automata-based characterisation of context-sensitive languages, as opposed to linearly bounded machines which are essentially a two-way mechanism. To prove the inclusion of context-sensitive languages in the set of path languages of these families of graphs, these papers use a normal form for growing grammars, due to Penttonen [Pen74]. In [CM06], these results were reformulated using simpler proof techniques based on tiling systems. This also allowed to investigate interesting sub-cases, in particular concerning deterministic context-sensitive languages or various sub-classes of rational graphs.

The aim of this work is to extend the results of [LS97] and [CM06] to the more general family ETIME of languages accepted by deterministic Turing machines working in time less than $2^{O(n)}$, or equivalently by alternating linearly bounded machines. This family lies between context-sensitive and recursively-enumerable languages in the Chomsky hierarchy. We obtain two new characterisations of ETIME, first as the languages accepted by arborescent tiling systems and second as the traces of infinite graphs defined by various classes of term transducers, namely term-synchronous and term-automatic (or tree-automatic) graphs [BG00].

After recalling definitions and notations in Section 1, we introduce the notion of arborescent tiling systems in Section 2 and prove that they characterise ETIME. Finally, we extend previously mentioned proofs over rational graphs to the family of term-automatic graphs in Section 3.

1 Notations

1.1 Words, terms and trees

A word u over alphabet Σ can be seen as a tuple $(a_1, \ldots, a_n)$ of elements of Σ, usually written $a_1 \ldots a_n$. Its i-th letter is denoted by $u(i) = a_i$. The set of all words over Σ is written Σ^*. The number of letters occurring in u is its length, written $|u|$ (here $|u| = n$). The empty word is written ε. The concatenation of two words $u = a_1 \ldots a_n$ and $v = b_1 \ldots b_m$ is the word $uv = a_1 \ldots a_n b_1 \ldots b_m$. The concatenation operation extends to sets of words: for all $A, B \subseteq \Sigma^*$, AB stands for the set $\{uv \mid u \in A \text{ and } v \in B\}$.

Let $F = \bigcup_{n \geq 0} F_n$ be a finite ranked alphabet, each F_n being the set of symbols of F of arity n, and X be a finite set of *variables* disjoint from F (all sets F_n are also disjoint). We denote the arity of a symbol $f \in F$ by $\mathrm{a}(f)$. Variables are considered of arity 0. The set of finite first-order terms on F with variables in X, written $T(F, X)$, is the smallest set including X such that $f \in F_n \ \wedge \ t_1, \ldots, t_n \in T(F, X) \Rightarrow f t_1 \ldots t_n \in T(F, X)$. Words can be seen as terms over a ranked alphabet whose symbols have arity exactly 1 and whose last symbol is a variable or a special constant. To improve readability, $f t_1 \ldots t_n$ will sometimes be written $f(t_1, \ldots, t_n)$.

A finite ordered tree t over a set of labels Σ is a mapping from a prefix-closed set $\mathrm{dom}(t) \subseteq \mathbb{N}^*$ into Σ. Elements of $\mathrm{dom}(t)$ are called positions, and for every $p \in \mathrm{dom}(t)$, $t(p)$ is the label of the node at position p. The node at position ε is called the root of the tree, nodes at maximal positions (i.e. positions x such that $\nexists y \neq \varepsilon,\ xy \in \mathrm{dom}(t)$) are called leaves, other nodes are called internal.

Any term t over a ranked alphabet F and set of variables X can be represented as a finite ordered ranked tree, whose leaves are labelled with constants in F_0 or variables in X and whose internal nodes are labelled with symbols of arity equal to the number of children of that node. In that case, the domain of t, additionally to being prefix-closed, also has the following properties:

1. $\forall p \in \mathrm{dom}(t),\ t(p) \in F_{n\geq 1} \implies \{j \mid pj \in \mathrm{dom}(t)\} = [1, n]$,
2. $\forall p \in \mathrm{dom}(t),\ t(p) \in F_0 \cup X \implies \{j \mid pj \in \mathrm{dom}(t)\} = \emptyset$.

In such a tree, position pi with $i \in \mathbb{N}$ always denotes the i-th child of node p. Conversely, any finite ordered tree t labelled over Σ can be represented as a ranked tree t', and hence as a term, by mapping each node label a to a set of symbols (a, n) in $\Sigma \times \mathbb{N}$, with $\mathrm{a}(a, n) = n$, and by renumbering all positions such that $\mathrm{dom}(t')$ verifies the above properties. This will usually be left implicit.

A finite tree (or term) automaton is a tuple $A = \langle Q, F, q_0, \delta\rangle$, where Q is a set of control states, F a ranked alphabet, q_0 the initial set and δ the set of transition rules of A of the form $(q, f, q_1, \ldots, q_n)$ with $\mathrm{a}(f) = n$. A run of A over a tree t is a mapping ρ from $\mathrm{dom}(t)$ to Q such that $\rho(\varepsilon) = q_0$ and for all node $u \in \mathrm{dom}(t)$ of arity $\mathrm{a}(u) = n$, $(\rho(u), t(u), \rho(u1), \ldots, \rho(un)) \in \delta$. If A has a valid run on t, we say that t is accepted by A. The set of trees accepted by a finite automaton is called its language, and all such languages are said to be *regular*.

1.2 Graphs

A labelled, directed and simple *graph* is a set $G \subseteq V \times \Sigma \times V$ where Σ is a finite set of labels and V an arbitrary countable set. An element (s, a, t) of G is an *edge* of *source* s, *target* t and *label* a, and is written $s \xrightarrow[G]{a} t$ or simply $s \xrightarrow{a} t$ if G is understood. An edge with the same source and target is called a *loop*. The set of all sources and targets of a graph form its *support* V_G, its elements are called *vertices*. A sequence of edges $(s_1 \xrightarrow{a_1} t_1, \ldots, s_k \xrightarrow{a_k} t_k)$ with $\forall i \in [2, k],\ s_i = t_{i-1}$ is called a *path*. It is written $s_1 \xrightarrow{u} t_k$, where $u = a_1 \ldots a_k$ is the corresponding *path label*. Vertex s_1 is called the origin of the path, t_k its destination. A path is called a *cycle* if its origin and destination are the same vertex. The language, or set of traces of a labelled graph between two sets I and F of vertices is the set of all words w such that there exists a path labelled by w whose origin is in I and destination in F.

1.3 Turing machines

A Turing machine is a tuple $M = \langle \Gamma, \Sigma, Q, q_0, F, \delta\rangle$ where Σ is the input alphabet, Γ the tape or work alphabet (with $\Sigma \subseteq \Gamma$), Q is a set of states among which q_0 is an initial state and F is a set of final states, and δ is a set of transition rules of the form $pA \to qB\epsilon$ where $p, q \in Q,\ A, B \in \Gamma \cup \{\Box\}$ ($\Box$ being a blank symbol not in Γ) and $\epsilon \in \{+, -\}$.

Configurations of M are denoted as words upv, where uv is the content of the work tape (where prefix and suffix blank symbols are omitted), p is the current control state and the head scans the cell containing the first letter of v. A transition $d = pA \to qB\epsilon$ is enabled on any configuration c of the form $upAv$, and yields a new configuration $d(c) = uBqv'$ (with $v' = v$ if $v \neq \varepsilon$, or $\Box$

otherwise) if $\epsilon = +$ and $u'qCBv$ (with $u'C = u$ if $u \neq \varepsilon$ or $u' = \varepsilon$ and $C = \Box$ otherwise) if $\epsilon = -$. If d is not enabled on c, then $d(c)$ is left undefined.

An *alternating* Turing machine M is defined similarly, with the exception that rules are of the form $d = pA \to \bigwedge_{i \in [1,n]} q_i B_i \epsilon_i$. The alternation degree n of d is written $\mathrm{a}(d)$, by analogy with the notion of arity. For all $i \leq \mathrm{a}(d)$, we write d_i the non-alternating transition $pA \to q_i B_i \epsilon_i$. A run of M on input word w is a tree whose root is labelled by configuration $q_0 w$, and such that the children of any node labelled by configuration c are labelled by $c_1, \ldots, c_n$ if and only if there exists a transition $d \in \delta$ enabled on c such that $\mathrm{a}(d) = n$ and $\forall i \in [1, n]$, $c_i = d_i(c)$. Such a run is successful if all its leaves are labelled by configurations whose control state is in F.

A Turing machine is *linearly bounded* if on every run the total work tape space it uses is at most proportional to the length of its input word. By standard coding techniques, it is sufficient to consider machines whose tape is limited to the cells initially containing the input word. This may be enforced by forbidding transition rules to rewrite the blank symbol $\Box$. The languages of non-alternating linearly bounded machines form the complexity class SPACE($O(n)$), which is equivalent to context-sensitive languages [Kur64]. Adding alternation, one obtains the more general class ASPACE($O(n)$). By classical complexity results [CKS81], it is also equivalent to the class DTIME($2^{O(n)}$), also called ETIME.

2 Arborescent tiling systems

To facilitate the proofs of our main results, this section provides an important technical tool, which was also central to some versions of the corresponding proofs on rational graphs and context-sensitive languages (Cf. [CM06]).

Tiling systems were originally defined to recognise or specify picture languages, i.e. sets of two-dimensional words on finite alphabets [GR96], called local picture languages. However, by only looking at the words contained in the first row of each picture of a local picture language, one obtains a context-sensitive language, and the converse is true : for any context-sensitive language there exists a local picture language (and a tiling system accepting it) whose set of upper frontiers is that language [LS97].

In this section, we extend this result to an arborescent extension of tiling systems, and prove that this new formalism characterises precisely the class ETIME.

2.1 Definitions

Instead of planar pictures, we consider so-called arborescent pictures, which are to ordinary pictures what terms are to words.

Definition 1 (Arborescent picture). *Let Γ be a finite alphabet, an arborescent picture p over Γ is a mapping from the set $X \times [1, m]$ to Γ, where $X \subseteq {\mathbb{N}_+}^*$ is a finite, prefix-closed set of sequences of positive integers (called positions in* *the framework of trees) and* m *is a positive integer called the width of* p*. The set* $\mathrm{dom}(p) = X \times [1, m]$ *is the domain of* p*. The set of arborescent pictures over* $X \times [1, m]$ *is written* $\mathrm{AP}(X, m)$*.*

Like in the case of trees, we assume that X is not only prefix-closed but also left-closed, i.e. $\forall i > 0,\ ui \in X \implies \forall j < i,\ uj \in X$. For a given picture $p \in \mathrm{AP}(X, m)$, we write $\mathrm{fr}(p)$ the word $w \in \Gamma^m$ such that $w(i) = p(\varepsilon, i)$, which we call the (upper) frontier of p.

Arborescent pictures of domain $X \times [1, m]$ are isomorphic to ordered trees of domain X with nodes labelled over the set Γ^m. As such, if $m = 1$ they are isomorphic to Γ-labelled ordered trees. One can observe that any branch of an arborescent picture seen as a Γ^m-labelled tree, as well as any arborescent picture whose set of positions X is a subset of 1^*, is an ordinary, planar picture.

Definition 2 (Sub-picture). *For any arborescent picture* $p \in \mathrm{AP}(X, m)$*, the sub-picture* $p' = p|_{x,i,Y,n}$ *of* p *at offset* $o = (x, i)$ *with* $x \in X$ *and* $i \in [0, m-1]$ *is the arborescent picture of domain* $Y \times [1, n]$ *such that* Y *is prefix- and left-closed and* $\forall (y, j) \in Y \times [1, n],\ (xy, i + j) \in X \times [1, m]$ *and* $p'(y, j) = p(xy, i + j)$*.*

We can now define arborescent tiling systems, which allow the specification of sets of arborescent pictures. Similarly to planar tiling systems, in order to be able to recognise meaningful sets of pictures, we first add a border or frame to each picture using a new symbol #.

Definition 3 (Framed picture). *Let* p *be an arborescent picture of domain* $X \times [1, m]$ *over* Γ *and* $\# \notin \Gamma$ *a new symbol, we define the #-framed picture* $p_\#$ *as the picture of domain* $X' \times [1, m+2]$ *with* $X' = \{\varepsilon\} \cup \{1\}X \cup X''$ *and* $X'' = \{1x1 \mid x \in X \wedge \nexists y \in \mathbb{N}, xy \in X\}$ *such that*

$$
\begin{aligned}
&p_\#(\varepsilon, i) = \# && \textit{for all } i \in [1, m+2],\\
&p_\#(1x, 1) = \# \textit{ and } p_\#(1x, m+2) = \# && \textit{for all } x \in X,\\
&p_\#(x, i) = \# && \textit{for all } x \in X'', i \in [1, m+2],\\
&p_\#(1x, i+1) = p(x, i) && \textit{for all } x \in X, i \in [1, m].
\end{aligned}
$$

An arborescent tiling system is then defined as a set of tiling elements of width and height 2, which can then be combined to form larger framed pictures.

Definition 4 (Arborescent tiling system). *An arborescent tiling system (or ATS)* S *is a triple* $(\Gamma, \#, \Delta)$*, where* Γ *is a finite alphabet,* $\# \notin \Gamma$ *a frame symbol and* Δ *is a set of arborescent tiling elements (tiles) in* $\{\bar{\Gamma} \times \bar{\Gamma} \times \bar{\Gamma}^n \times \bar{\Gamma}^n \mid n > 0\}$ *with* $\bar{\Gamma} = \Gamma \cup \{\#\}$*.*

Each tiling element $d \in \Delta$ is of the form $d = (A, B, \bar{C}, \bar{D})$ with $A, B \in \bar{\Gamma}$ and $\bar{C}, \bar{D} \in \bar{\Gamma}^n$ for some positive integer n. We define additional notations to conveniently manipulate tiling elements. Let $d = (A, B, \bar{C}, \bar{D})$ with $\bar{C} = C_1 \dots C_n$ and $\bar{D} = D_1 \dots D_n$, we write $\mathrm{a}(d) = n$ to denote the arity of d, and d_i with $i \in [1, \mathrm{a}(d)]$ to denote the (planar) tile (A, B, C_i, D_i).

Note that any tiling element $d = (A, B, \bar{C}, \bar{D})$ of arity n is isomorphic to an arborescent picture p_d of domain $X \times [1, 2]$, where $X = \{\varepsilon, 1, \ldots, n\}$ and $p_d(\varepsilon, 1)$, $p_d(\varepsilon, 2)$, $p_d(i, 1)$ and $p_d(i, 2)$ are respectively equal to A, B, C_i and D_i (for all $i \in [1, n]$). In general we do not distinguish p_d from d and write simply d.

Well-formed tiling systems should obey a certain number of restrictions over their set of tiles, regarding in particular the occurrences of the frame symbol # inside tiles. For all $d = (A, B, \bar{C}, \bar{D})$,

1. $(A, B) = (\#, \#) \implies \mathrm{a}(d) = 1 \wedge (C_1, D_1) \neq (\#, \#)$,
2. $\exists i, (C_i, D_i) = (\#, \#) \implies \mathrm{a}(d) = 1 \wedge (A, B) \neq (\#, \#)$,
3. $\exists i, C_i = \# \wedge D_i \neq \# \implies A = \# \wedge \forall i, C_i = \#$,
4. $\exists i, D_i = \# \wedge C_i \neq \# \implies B = \# \wedge \forall i, D_i = \#$,

Before defining the set of pictures and the word language accepted by an arborescent tiling system, we define for any arborescent picture p of domain $X \times [1, m]$ over Γ the set $\mathrm{T}(p)$ of tiling elements of p as the set of all sub-pictures $p|_{x,j,X',2}$ of p such that x is an internal position in X, $j \in [1, m-1]$ and $X' = \{\varepsilon\} \cup \{i' > 0 \mid xi' \in X\}$.

Definition 5 (Language of a tiling system). *The set of arborescent pictures accepted by an arborescent tiling system* $S = (\Gamma, \#, \Delta)$ *is the set* $\mathrm{P}(S) = \{p \in \mathrm{AP} \mid \mathrm{T}(p_\#) \subseteq \Delta\}$. *The (word) language accepted by* S *is the set* $\mathrm{L}(S) = \{w \in \Gamma^* \mid \exists p \in \mathrm{P}(S), w = \mathrm{fr}(p)\}$ *of all upper frontiers of pictures of* $\mathrm{P}(S)$.

As previously, note that arborescent tiling systems are a syntactical generalisation of planar tiling systems : framed pictures with a domain $X \subseteq 1^*$ or branches of framed arborescent pictures are planar framed pictures, and arborescent tiling systems whose elements all have arity 1 are ordinary, planar tiling systems.

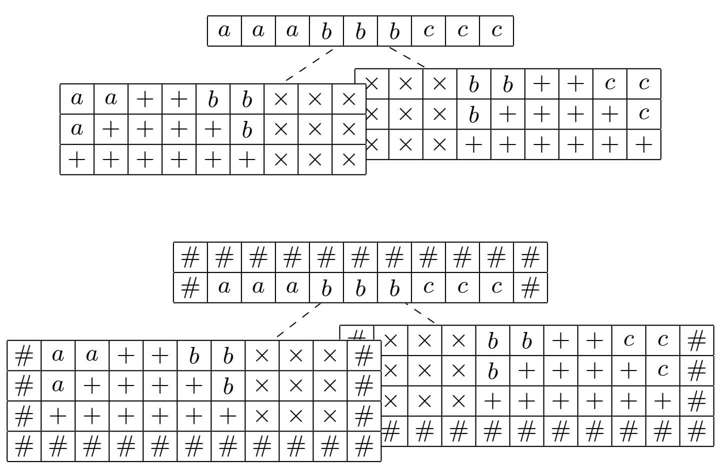

Fig. 1. Arborescent picture p and the corresponding framed picture $p_\#$

Example 1. Figure 2.1 represents an arborescent picture p whose frontier is the word $a^3b^3c^3$, as well as the corresponding framed picture. For the sake of clarity, the tree-structure of p is denoted by dashed lines only where the arity is greater than 1. By considering all sub-pictures of height and width 2 of that framed picture, one obtains a set of tiling elements Δ, which contains, among others, tiling elements $(\#, \#, a, b)$, $(a, +, +, +)$ and $(+, +, \#, \#)$ of arity 1 and $(b, c, b+, \times+)$ of arity 2, but not $(b, c, b+, c+)$ or $(\#, +, \#, +)$ for instance.

One can see that the tiling system $S = (\{a, b, c, +, \times\}, \#, \Delta)$ accepts all arborescent pictures similar to p whose frontiers are words of the form $a^n b^n c^n$ with $n \geq 2$: the left branch of each such picture ensures that the number of a's and b's is equal by replacing at each successive row one occurrence of a and one occurrence of b by some symbol $+$. Occurrences of c are irrelevant and are replaced with symbol $\times$. A lower frame borders can only be generated once all occurrences of a and b have been replaced. A similar check is performed by the right branch for symbols b and c.

Note that S does not accept the word abc, since accepting a similar picture with frontier abc would require some additional tiling elements, like for instance $(a, b, +\times, ++)$ and $(b, c, ++, \times+)$. Consequently, the language $\mathrm{L}(S)$ is $\{a^n b^n c^n \mid n \geq 2\}$.

2.2 Languages of arborescent tiling systems

In this section, we prove that arborescent tiling systems and alternating linearly bounded machines define the same family of languages, namely ASPACE$(O(n))$, also equal as previously mentioned to DTIME$(2^{O(n)})$ = ETIME.

Proposition 1. *For every arborescent tiling system S, there exists an alternating linearly bounded machine M such that* $\mathrm{L}(M) = \mathrm{L}(S)$.

Proof sketch. Let $S = (\Gamma, \#, \Delta)$ be an arborescent tiling system. We build an alternating linearly bounded machine $M = (\Gamma, \Gamma', Q, q_\#, f, \delta)$ accepting $\mathrm{L}(S)$. Its work alphabet Γ' is the union of all $\bar{\Gamma}^k$ for $k \in [1, \mathrm{a}(S)]$, where $\bar{\Gamma} = \Gamma \cup \{\#\}$ and $\mathrm{a}(S) = \max\{\mathrm{a}(d) \mid d \in \Delta\}$. We informally describe M's behaviour as follows:

1. M starts in configuration $[q_\# w]$, where $w \in \Gamma^*$ is the input word. In a first sweep, it checks that w is a possible frontier of a picture accepted by S.
2. In the next sweep, M generates a n-tuple of possible next rows based on the current configuration and the tiles in Δ. M then uses universal branching to evaluate the sub-pictures whose upper frontiers are each of these rows.
3. The last generated row consists in a sequence of frame symbols $\#$ if and only if the last written symbol is $\#$. If this is the case on the current computation branch, reach accepting state f. Otherwise, repeat the previous step.

Steps 2 and 3 are repeated until all computation branches have reached the accepting state f. □

Proposition 2. *For every alternating linearly bounded machine M, there exists an arborescent tiling system S such that* $\mathrm{L}(S) = \mathrm{L}(M)$.

Proof sketch. Let $M = (\Sigma, \Gamma, Q, q_0, F, \delta)$ be an alternating linearly bounded machine. We build an arborescent tiling system $S = (\Gamma', \#, \Delta)$ such that $\mathrm{L}(S) = [\mathrm{L}(M)]$, where [and] are two new symbols. The set of tiling elements Δ is built in order to conform to the following informal specification.

S first needs to set an input word w as the upper frontier of any picture it accepts. It then encodes the initial configuration of M on w as the second row. Subsequent tiles simulate the application of a transition of M on the configuration encoded by the current row, and check that the previous transition was correctly simulated. This requires additional information, in particular about the position of the head and the index of the last simulated transition, to be added to the picture alphabet. Arity n tiling elements are used when the simulated rule is of alternation degree n. This process goes on until an accepting state is reached by M on a given execution branch. In that case, a bottom border is generated by S on the corresponding picture branch. □

From Propositions 1 and 2, we deduce the announced theorem.

Theorem 1. *The languages of arborescent tiling systems form the complexity class* ETIME*.*

Note that the language accepted by the tiling system of Example 1 is a context-sensitive language, which could also be accepted by a non-arborescent tiling system.

3 Traces of term-automatic graphs

We now turn to the main result of this paper, which is the study of languages of graphs characterised by automata-defined binary relations over terms, and in particular term-automatic graphs. We define these relations and the graphs they generate, then present a two-steps proof that the languages of term-automatic graphs indeed coincide with ASPACE($O(n)$). First, we establish this result for the simpler term-synchronous graphs in Section 3.2, then generalise it to term-automatic graphs in Section 3.3.

3.1 Definitions

Let $s = f(s_1 \ldots s_m)$ and $t = g(t_1 \ldots t_n)$ be two terms over some ranked alphabet F. We define the overlap $[st]$ of s and t as a term over domain $\mathrm{dom}(s) \cup \mathrm{dom}(t)$ and extended alphabet $(F \cup \{\perp\})^2$ (each element (f, g) of this alphabet being written simply fg), such that $\forall p \in \mathrm{dom}(s) \cup \mathrm{dom}(t)$, $[st](p) = fg$ with $f = s(p)$ if $p \in \mathrm{dom}(s)$ or $\perp$ otherwise, and $g = t(p)$ if $p \in \mathrm{dom}(t)$ or $\perp$ otherwise. This notation is extended to sets in the natural way.

We can now define term-automatic and term-synchronous relations. We say a binary relation R is *term-* (or *tree-*)*automatic* if the term language $[R] = \{[st] \mid (s,t) \in R\}$ is regular. If furthermore for all $(s,t) \in R$, $\mathrm{dom}(s) = \mathrm{dom}(t)$, it is called *synchronous*. In other words, a synchronous relation is an automatic relation which only associates terms with the same domain. Both families of relations are closed under relational composition. Term-automatic and term-synchronous relations are syntactical extensions of the corresponding families of relations over words. As such, they also define extended families of graphs.

Definition 6. *A Σ-graph G is term-automatic (resp. term-synchronous) if it is isomorphic to a graph $\{u \xrightarrow{a} v \mid a \in \Sigma,\ (u, v) \in R_a\}$, where $(R_a)_{a\in\Sigma}$ is a family of term-automatic (resp. term-synchronous) relations.*

3.2 Term-synchronous graphs

This section presents direct simulations of alternating tiling systems by synchronous graphs and conversely, showing that the languages of term-synchronous graphs between regular sets of vertices form the class ETIME.

Proposition 3. *For every term-synchronous graph G and regular sets I and F there exists an arborescent tiling system S such that* $\mathrm{L}(S) = \mathrm{L}(G, I, F)$.

Proof sketch. Let $G = (R_a)_{a\in\Sigma}$ be a synchronous graph, and I, F two regular sets of vertices of G. We build a tiling system $S = (\Gamma, \#, \Delta)$ such that $\mathrm{L}(S) = \mathrm{L}(G, I, S)$.

For all $a \in \Sigma$, let A_a be a finite top-down term automaton accepting the language $[R_a]$ (as defined in Section 3.1), and A_I, A_F similar automata for I and F respectively. For every $a \in \Sigma$, we also define relations $R_{I\circ a} = Id_I \circ R_a$ and $R_{a\circ F} = R_a \circ Id_F$, where Id_L denotes the identity relation over some set L. Let also $A_{I\circ a}$ and $A_{a\circ F}$ be two automata accepting the languages $[R_{I\circ a}]$ and $[R_{a\circ F}]$ respectively. The control state sets of all these automata are supposed disjoint.

The idea of this construction is that, for every path $t_0 \xrightarrow{a_1} t_1 \ldots \xrightarrow{a_n} t_n$ in G with $t_0 \in I$, $t_n \in F$ and $\forall i,\ \mathrm{dom}(t_i) = X$, S should accept an arborescent picture p whose upper frontier is w and whose successive vertical "slices" correspond to encodings of runs of $A_{I\circ a_1}$, A_{a_2}, $\ldots$, $A_{a_{n-1}}$ and $A_{a_n\circ F}$ respectively. Conversely, S should only accept all such pictures which correspond to paths in G between I and F. These conditions are sufficient for $L(S)$ to be equal to $\mathrm{L}(G, I, F)$. To ensure they indeed hold, we define Δ in order to be able to check that the i-th and $(i + 1)$-th "slices" are indeed compatible. $\square$

Proposition 4. *For every arborescent tiling system S, there exists a term-synchronous graph G and regular sets I and F such that* $\mathrm{L}(G, I, F) = \mathrm{L}(S)$.

Proof sketch. Let $S = (\Gamma, \#, \Delta)$ be an arborescent tiling system. We build a term-synchronous graph G such that $\mathrm{L}(S) = \mathrm{L}(G, I, F)$ for some regular sets I and F. In the following, symbol # is overloaded to make the notation less cumbersome, and represents functional symbols of varying arities, which can be deduced from the context. In particular, we write $\#_X$ for a given prefix-closed set X the term of domain X whose nodes are all labelled with #.

Let R_a, $a \in \Sigma$, be the binary relation between all terms $\#(s)$ and $\#(t)$ (i.e. s and t with an additional unary # at the root) such that a labels the root of t and for a given $p \in P(S)$, either $s = p|_{\varepsilon,i,X,1}$ and $t = p|_{\varepsilon,i+1,X,1}$ for some $i > 0$ or $s = \#_X$ and $t = p|_{\varepsilon,0,X,1}$. Let G be the graph defined by $(R_a)_{a\in\Sigma}$, we show that G is term-synchronous by constructing automata $(A_a)_{a\in\Sigma}$ such that $\mathrm{L}(A_a) = [R_a] = \{[st] \mid (s,t) \in R_a\}$. For all a, A_a has transitions:

$$
\begin{array}{ll}
q_0\#\# \to q_{AB,1} & \text{if } (\#,\#,A,B) \in \Delta, \\
q_{\bar{A}\bar{B},i}AB \to q_{\bar{C}\bar{D},1} \ldots q_{\bar{C}\bar{D},k} & \text{if } d = (A_i,B_i,\bar{C},\bar{D}) \in \Delta,\ k = \mathrm{a}(d), \\
 & \quad A_i = \bar{A}(i) \text{ and } B_i = \bar{B}(i), \\
q_{\bar{A}\bar{B},i}A_iB_i \to \varepsilon & \text{if } (A_i,B_i,\#,\#) \in \Delta, \\
 & \quad A_i = \bar{A}(i) \text{ and } B_i = \bar{B}(i).
\end{array}
$$

We define I as the regular set of all terms labelled over $\{\#\}$, and F as the set of all possible rightmost columns of pictures accepted by S. This set of terms is accepted by an automaton A_F whose construction is straightforward.

By construction of I, A_F and each of the A_a, there is a path in G labelled by a word w between a vertex in I and a vertex in F iff the vertices along that path are the successive columns of a picture in $P(S)$ with frontier w. $\square$

3.3 Term-automatic graphs

In this section, we show that the more general family of term-automatic graphs defines the same family of languages as their synchronous counterparts.

Proposition 5. *For every term-automatic graph G and regular sets of terms I and F, there exists a term-synchronous graph G' and regular sets I' and F' such that* $\mathrm{L}(G', I', F') = \mathrm{L}(G, I, F)$.

Proof sketch. Let G be a term-automatic graph defined by a family $(R_a)_{a\in\Sigma}$ of automatic relations and I, F be two regular languages, each $[R_a]$ being accepted by an automaton A_a, I by A_I and F by A_F. We define a synchronous graph $G' = (R'_a)_{a\in\Sigma}$ and two regular sets I' and F' such that $\mathrm{L}(G, I, F) = \mathrm{L}(G', I', F')$.

Recall that term-automatic relations are defined using a notion of overlap between terms (Cf. Section 3.1). Two terms s and t with different domains belong to a term-automatic relation R defined by automaton A if the overlap $[st]$ of s and t is accepted by A. This notion of overlap consists in "unifying" the domains of s and t, and padding undefined positions with a special symbol $\perp$.

We wish to reuse this idea, but instead of unifying the domains of two terms only, we have to unify the domains of all vertices along a given path. Indeed, in a term-synchronous graph, edges can only exist between terms with precisely the same domain. For every term s standing for a vertex in G, we will thus have to consider an infinite set of "versions" of s in G', one for each possible term domain larger than that of s.

Let Γ be a ranked alphabet, we define alphabet Γ' as $\Gamma' = \Gamma_0 \cup \Gamma_n$ with $\Gamma'_0 = \#_0$ and $\Gamma'_n = \Gamma \cup \#_n$, where n is the maximal arity of symbols in Γ. Let

ϕ be a mapping from $T(\Gamma)$ to $2^{T(\Gamma')}$ such that for any term $t \in T(\Gamma)$,

$$\phi(t) \;=\; \big\{t' \in T(\Gamma') \mid \mathrm{dom}(t) \subset \mathrm{dom}(t'),\ \forall p \in \mathrm{dom}(t), t'(p) = t(p) \\ \text{and } \forall p \in \mathrm{dom}(t') \setminus \mathrm{dom}(t), t'(p) \in \{\#_0, \#_n\}\big\}.$$

In other words, to any term t, ϕ associates the set of all terms obtained by "padding" t with silent symbols $\#_0$ and $\#_n$. This mapping is extended to sets of terms in the natural way. Note that, given any $t' \in F(\Gamma')$, there exists at most one term $t \in T(\Gamma)$ such that $t' \in \phi(t)$.

We now define, for every $a \in \Sigma$, relation R'_a as $\{(s', t') \mid (s, t) \in R_a,\ s' \in \phi(s),\ t' \in \phi(t) \text{ and } \mathrm{dom}(s') = \mathrm{dom}(t')\}$. This synchronous relation can be characterised by a finite tree automaton A'_a defined from A_a. We also let $I' = \phi(I)$ and $F' = \phi(F)$, for which automata can be similarly defined from A_I and A_F.

Let G' be the term-synchronous graph defined by $(R'_a)_{a\in\Sigma}$. One can show that for every path labelled w in G' between some $i' \in I'$ and $f' \in F'$, there exists a unique path between I and F in G with the same label, and that conversely for every w-path between I and F in G there must exist at least one corresponding path in G' between I' and F'. This ensures that $L(G, I, F)$ and $L(G', I', F')$ are indeed equal. □

Remark 1. Note that for every term-automatic graph G and regular sets I and F, there exists a term-automatic graph G' and finite sets I' and F' such that $\mathrm{L}(G', I', F') = \mathrm{L}(G, I, F)$. Indeed, for any regular I and F and finite I' and F' the relations $I' \times I$ and $F \times F'$ are automatic. Since term-automatic relations are closed under union and composition, this can be used to build G' from G.

This, however, does not hold in the term-synchronous case. Indeed, since each connected component of a term-synchronous graph is finite, the language of any such graph from a finite set of initial vertices is regular.

Combining Theorem 1, Propositions 3, 4 and 5, as well as Remark 1, we obtain the following result concerning the family of languages accepted by term-synchronous and term-automatic graphs.

Theorem 2. *The languages of term-synchronous graphs between regular sets of vertices and of term-automatic graphs between regular or finite sets of vertices form the complexity class* ETIME.

4 Conclusion

We have proved that the class of languages accepted by alternating linearly bounded machines (ETIME) can also be characterised as the sets of first rows of pictures accepted by arborescent tiling systems, as well as the sets of path labels of term-automatic graphs between regular or finite sets of initial and final vertices.

A natural extension of this work would be to generalise Theorem 2 to graphs defined by more expressive classes of tree transducers, in order to fully extend the existing results on rational graphs. In practice, this would require extending the construction for Proposition 5 to more general padding techniques.

Further points of interest concern the extension of other results from [CM06] to term-automatic graphs, in particular regarding structural restrictions of these graphs, like finite or bounded degree, or the restriction to a single initial vertex, as well as a similar study of related complexity classes or families of languages.

References

[BG00] A. Blumensath and E. Grädel. Automatic structures. In *Proceedings of the 15th IEEE Symposium on Logic in Computer Science (LICS 2000)*, pages 51–62. IEEE, 2000.

[Cho59] N. Chomsky. On certain formal properties of grammars. *Information and Control*, 2:137–167, 1959.

[CKS81] A. Chandra, D. Kozen, and L. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, 1981.

[CM06] A. Carayol and A. Meyer. Context-sensitive languages, rational graphs and determinism. *Logical Methods in Computer Science*, 2(2), 2006.

[GR96] D. Giammarresi and A. Restivo. *Handbook of Formal Languages*, volume 3, chapter Two-dimensional languages. Springer, 1996.

[KN95] B. Khoussainov and A. Nerode. Automatic presentations of structures. In *International Workshop on Logical and Computational Complexity (LCC '94)*, pages 367–392. Springer, 1995.

[Kur64] S. Kuroda. Classes of languages and linear-bounded automata. *Information and Control*, 7(2):207–223, 1964.

[LS97] M. Latteux and D. Simplot. Context-sensitive string languages and recognizable picture languages. *Information and Computation*, 138(2):160–169, 1997.

[Mor00] C. Morvan. On rational graphs. In *Proceedings of the 3rd International Conference on Foundations of Software Science and Computation Structures (FoSSaCS 2000)*, volume 1784 of *Lecture Notes in Computer Science*, pages 252–266. Springer, 2000.

[MR05] Christophe Morvan and Chloé Rispal. Families of automata characterizing context-sensitive languages. *Acta Informatica*, 41(4-5):293–314, 2005.

[MS01] C. Morvan and C. Stirling. Rational graphs trace context-sensitive languages. In *Proceedings of the 26th International Symposium on Mathematical Foundations of Computer Science (MFCS 2001)*, volume 2136 of *Lecture Notes in Computer Science*, pages 548–559. Springer, 2001.

[Pen74] M. Penttonen. One-sided and two-sided context in formal grammars. *Information and Control*, 25(4):371–392, 1974.

[Ris02] C. Rispal. The synchronized graphs trace the context-sensitive languages. In *Proceedings of the 4th International Workshop on Verification of Infinite-State Systems (INFINITY 2002)*, volume 68 of *Electronic Notes in Theoretical Computer Science*, 2002.